Date=5/11/2020
Lecture By=Manish Mahant
Subject ⇒Let And Const Destructuring

IN PREVIOUS LECTURE **(QUICK RECAP) Date-4/11/2020**	**In Today's Lecture (Overview)**
Babel is a JavaScript compiler Usage Guide Overview Core Library Configuration What Is Lodash In Javascript Installation Why Lodash? Questions For Self-practice // Assignment For The Day	JavaScript Let JavaScript Const Destructuring assignment Syntax Description Question For Self-practice // CC For The Day

JavaScript Let

ECMAScript 2015

ES2015 introduced two important new JavaScript keywords: `let` and `const`.

These two keywords provide Block Scope variables (and constants) in JavaScript.

Before ES2015, JavaScript had only two types of scope: Global Scope and Function Scope.

Global Scope

Variables declared Globally (outside any function) have Global Scope.

Example

```
var carName = "Volvo";

// code here can use carName

function myFunction() {

  // code here can also use carName

}
```

Function Scope

Variables declared Locally (inside a function) have Function Scope

Example

```
// code here can NOT use carName

function myFunction() {

  var carName = "Volvo";

  // code here CAN use carName

}

// code here can NOT use carName
```

JavaScript Block Scope

Variables declared with the `var` keyword cannot have Block Scope.

Variables declared inside a block {} can be accessed from outside the block.

Example

```
{
  var x = 2;
}
// x CAN be used here
```

JavaScript Const

ECMAScript 2015

ES2015 introduced two important new JavaScript keywords: `let` and `const`.

Variables defined with `const` behave like `let` variables, except they cannot be reassigned:

Example

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

Block Scope

Declaring a variable with `const` is similar to `let` when it comes to Block Scope.

The x declared in the block, in this example, is not the same as the x declared outside the block:

Example

```
var x = 10;

// Here x is 10

{

  const x = 2;

  // Here x is 2

}

// Here x is 10
```

Assigned when Declared

JavaScript `const` variables must be assigned a value when they are declared:

Incorrect

```
const PI;

PI = 3.14159265359;
```

Correct

```
const PI = 3.14159265359;
```

Constant Arrays can Change

You can change the elements of a constant array:

Example

```
// You can create a constant array:

const cars = ["Saab", "Volvo", "BMW"];



// You can change an element:

cars[0] = "Toyota";



// You can add an element:

cars.push("Audi");
```

Destructuring assignment

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

```
let a, b, rest;
[a, b] = [10, 20];

console.log(a);
// expected output: 10
```

```
console.log(b);
// expected output: 20

[a, b, ...rest] = [10, 20, 30, 40, 50];

console.log(rest);
// expected output: Array [30,40,50]
```

Syntax

```
let a, b, rest;
[a, b] = [10, 20];
console.log(a); // 10
console.log(b); // 20

[a, b, ...rest] = [10, 20, 30, 40, 50];
console.log(a); // 10
console.log(b); // 20
console.log(rest); // [30, 40, 50]

({ a, b } = { a: 10, b: 20 });
console.log(a); // 10
console.log(b); // 20


// Stage 4(finished) proposal
({a, b, ...rest} = {a: 10, b: 20, c: 30, d: 40});
console.log(a); // 10
console.log(b); // 20
console.log(rest); // {c: 30, d: 40}
```

Description

The object and array literal expressions provide an easy way to create *ad hoc* packages of data

The destructuring assignment uses similar syntax, but on the left-hand side of the assignment to define what values to unpack from the sourced variable.

```
const x = [1, 2, 3, 4, 5];
const [y, z] = x;
console.log(y); // 1
console.log(z); // 2
```

Assignment without declaration

A variable can be assigned its value with destructuring separate from its declaration.

```
let a, b;

({a, b} = {a: 1, b: 2});
```

Assigning to new variable names

A property can be unpacked from an object and assigned to a variable with a different name than the object property.

```
const o = {p: 42, q: true};
const {p: foo, q: bar} = o;

console.log(foo); // 42
console.log(bar); // true
```

Unpacking fields from objects passed as a function parameter

```
const user = {
  id: 42,
  displayName: 'jdoe',
```

```
  fullName: {
    firstName: 'John',
    lastName: 'Doe'
  }
};

function userId({id}) {
  return id;
}

function whois({displayName, fullName: {firstName: name}}) {
  return `${displayName} is ${name}`;
}

console.log(userId(user)); // 42
console.log(whois(user));  // "jdoe is John"
```

To Know More about It Visit The link Below
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Question For Self-practice // CC For The Day

https://au-assignment.s3.ap-south-1.amazonaws.com/Week_19_Day_4_Challenge-8bca30ee-e5f6-4062-9d4a-7878f3070f0c.pdf